

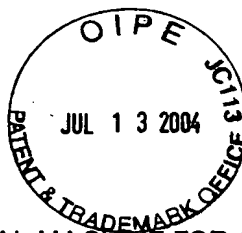
IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Venkatesh Krishnan et al.

Application No.: 09/264,756

Filing Date: 3-9-99

Title: CLASS LOADING IN A VIRTUAL MACHINE FOR A PLATFORM HAVING MINIMAL RESOURCES



Confirmation No.: 3679

Examiner: Nguyen D.

Group Art Unit: 2154

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

RECEIVED

JUL 19 2004

Technology Center 2100

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith in **triplicate** is the Appeal Brief in this application with respect to the Notice of Appeal filed on 1-20-04.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$330.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

() (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

| | |
|------------------|-----------|
| () one month | \$110.00 |
| () two months | \$420.00 |
| () three months | \$950.00 |
| () four months | \$1480.00 |

() The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of \$330.00. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:
Commissioner for Patents, Alexandria, VA
22313-1450. Date of Deposit: 7-9-04

OR

() I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number _____ on _____

Number of pages:

Typed Name: Paul H. Horstmann

Signature: Paul H. Horstmann

Respectfully submitted,

Venkatesh Krishnan et al.

By Paul H. Horstmann

Paul H. Horstmann

Attorney/Agent for Applicant(s)

Reg. No. 36,167

Date: 7-9-04

Telephone No.: (310) 376-0218



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re Application of:

Venkatesh Krishnan et al.

Application No: 09/264,756

Filed: 3-9-99

For: CLASS LOADING IN A VIRTUAL
MACHINE FOR A PLATFORM
HAVING MINIMAL RESOURCES

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Examiner: Nguyen

Art Unit: 2154

RECEIVED

JUL 19 2004

Technology Center 2100

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on

7-9-04

Date of Deposit

Paul H. Horstmann

Name of Person Mailing Correspondence

Paul H. Horstmann

Signature

7-9-04

Date

Appellant's Brief (Pursuant to 37 C.F.R. §1.192)

Dear Sir:

In response to the Notification of Non-Compliance dated 6-9-04, Applicant/Appellant submits this Appeal Brief in connection with the above-referenced patent application which is on appeal to the Board of Patent Appeals and Interferences.

07/15/2004 HALI11 00000053 082025 09264756

01 FC:1402 330.00 DA

Appeal Brief
Application Ser. No.: 09/264,756

1

Attorney Docket No.: 10981459-1

TABLE OF CONTENTS

| | |
|---|-----------|
| REAL PARTY IN INTEREST | 3 |
| RELATED APPEALS AND INTERFERENCES | 3 |
| STATUS OF THE CLAIMS | 3 |
| STATUS OF AMENDMENTS | 3 |
| SUMMARY OF THE INVENTION | 4 |
| ISSUES PRESENTED | 4 |
| I: WHETHER CLAIMS 20-22, 25-28, 31-32, 35-37, 40-41, AND 44-46 ARE OBVIOUS IN VIEW OF BAK AND EBRAHIM. | 4 |
| II: WHETHER CLAIMS 23-24, 33-34, AND 42-43 ARE OBVIOUS IN VIEW OF BAK AND EBRAHIM AND SHAUGHNESSY. | 4 |
| III: WHETHER CLAIMS 29-30 AND 38-39 ARE OBVIOUS IN VIEW OF BAK AND EBRAHIM AND BROWN. | 4 |
| GROUPING OF CLAIMS | 4 |
| ARGUMENT | 5 |
| I: CLAIMS 20-22, 25-28, 31-32, 35-37, 40-41, AND 44-46 ARE NOT OBVIOUS IN VIEW OF BAK AND EBRAHIM BECAUSE BAK AND EBRAHIM DO NOT DISCLOSE OR SUGGEST THE LIMITATIONS OF CLAIMS 20, 31, AND 40. | 5 |
| <i>A. Bak and Ebrahim do not disclose or suggest obtaining a set of classes via a network as needed while executing an application program as claimed in claims 20, 31, and 40.</i> | <i>5</i> |
| <i>B. Bak and Ebrahim do not disclose or suggest selecting and purging arrays and references from a class structure as claimed in claims 20, 31, and 40.</i> | <i>6</i> |
| <i>C. Bak and Ebrahim do not disclose or suggest selecting and purging arrays and references from a class structure so as to minimize an amount of memory consumed by the class structure while also minimizing class loading activities on a network as claimed in claims 20, 31, and 40.</i> | <i>7</i> |
| II: CLAIMS 23-24, 33-34, AND 42-43 ARE NOT OBVIOUS IN VIEW OF BAK AND EBRAHIM AND SHAUGHNESSY BECAUSE BAK AND EBRAHIM AND SHAUGHNESSY DO NOT DO NOT DISCLOSE OR SUGGEST THE LIMITATIONS OF CLAIMS 20, 31, AND 40. | 9 |
| III: CLAIMS 29-30 AND 38-39 ARE NOT OBVIOUS IN VIEW OF BAK AND EBRAHIM AND BROWN BECAUSE BAK AND EBRAHIM AND BROWN DO NOT DO NOT DISCLOSE OR SUGGEST THE LIMITATIONS OF CLAIMS 20 AND 31. | 10 |
| CONCLUSION | 11 |
| APPENDIX | 12 |

REAL PARTY IN INTEREST

The real party in interest in this application is Hewlett-Packard Development Company, L.P.

RELATED APPEALS AND INTERFERENCES

Appellant is unaware of any other related appeals or interferences that may directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

STATUS OF THE CLAIMS

Claims 20-22, 25-28, 31-32, 35-37, 40-41, and 44-46 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 5,999,732 of *Bak et al.* ("*Bak*") in view of U.S. Patent No. 5,848,423 of *Ebrahim et al.* ("*Ebrahim*").

Claims 23-24, 33-34, and 42-43 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Bak* and *Ebrahim* and U.S. Patent No. 5,787,431 of *Shaughnessy* ("*Shaughnessy*").

Claims 29-30 and 38-39 stand rejected under 35 U.S.C. §103(a) as being unpatentable over *Bak* and *Ebrahim* and U.S. Patent No. 6,295,643 of *Brown et al.* ("*Brown*").

Appellant appeals the rejection of all of the pending claims 20-46. Claims 20-46 as currently pending are set forth in the attached Appendix.

STATUS OF AMENDMENTS

Appellant is unaware of any amendments filed after the Final Office Action mailed 10/20/03 which finally rejected claims 20-46.

SUMMARY OF THE INVENTION

Claims 20-46 are directed to class loading in a virtual machine for a platform having limited resources. A virtual machine according to claims 20-46 obtains a set of classes via a network as needed while executing an application program (Appellant's Specification, page 5, lines 12-19), thereby minimizing the use of limited storage resources that would otherwise be consumed by storing the classes when the classes are not needed to execute the application program (Appellant's Specification, page 3, lines 3-17 and page 7, lines 6-18). A virtual machine according to claims 20-46 stores the obtained classes as arrays and references in a class structure (22 of Figure 1) and then selects and purges the arrays and references so as to minimize an amount of the memory consumed by the class structure while also minimizing class loading activities on the network (Appellant's Specification, Figure 3 and page 3, lines 3-17 and page 7, lines 10-18), thereby balancing the use the limited storage resource versus network resources.

ISSUES PRESENTED

I: Whether claims 20-22, 25-28, 31-32, 35-37, 40-41, and 44-46 are obvious in view of *Bak* and *Ebrahim*.

II: Whether claims 23-24, 33-34, and 42-43 are obvious in view of *Bak* and *Ebrahim* and *Shaughnessy*.

III: Whether claims 29-30 and 38-39 are obvious in view of *Bak* and *Ebrahim* and *Brown*.

GROUPING OF CLAIMS

Claims 20-46 stand together (Group I).

ARGUMENT

I: Claims 20-22, 25-28, 31-32, 35-37, 40-41, and 44-46 are not obvious in view of *Bak* and *Ebrahim* because *Bak* and *Ebrahim* do not disclose or suggest the limitations of claims 20, 31, and 40.

Appellant respectfully submits that claims 20, 31, and 40, and claims 21-22, 25-28, 32, 35-37, 41, and 44-46, are not obvious under 35 U.S.C. §103 in view of *Bak* and *Ebrahim* because *Bak* and *Ebrahim* do not teach or suggest the limitations in claims 20, 31, and 40 that are directed to class loading in a virtual machine having limited resources. *Bak* and *Ebrahim* do not disclose or suggest obtaining a set of classes via a network as needed while executing an application program as claimed in claims 20, 31, and 40. *Bak* and *Ebrahim* do not disclose or suggest selecting and purging arrays and references from a class structure as claimed in claims 20, 31, and 40. *Bak* and *Ebrahim* do not disclose or suggest purging arrays and references from a class structure so as to minimize an amount of memory consumed by the class structure while also minimizing class loading activities on a network as claimed in claims 20, 31, and 40.

A. *Bak* and *Ebrahim* do not disclose or suggest obtaining a set of classes via a network as needed while executing an application program as claimed in claims 20, 31, and 40.

Appellant submits that *Bak* and *Ebrahim* do not disclose or suggest obtaining a set of classes via a network as needed while executing an application program as claimed in claims 20, 31, and 40. *Bak* discloses a Java runtime system 201 that loads a set of Java class files 203 (*Bak*, col. 5, line 51 through col. 6, line 3) but does not teach or suggest that the Java class files 203 are loaded via a network as claimed in claims 20, 31, and 40. Figure 4 of *Bak* shows the Java class files 203 as an input to a Java runtime system 201 without any indication of network access. Similarly, *Ebrahim* discloses a class loader 146 that loads classes into a user's address space (*Ebrahim*, col. 3, lines 27-30)

but does not teach or suggest that the class loader 146 obtains classes via a network as claimed in claims 20, 31, and 40.

The Examiner has stated in support of his assertion that *Bak* teaches loading class files via a network that

Bak discloses a computer readable storage medium includes [sic] system memory and a hard drive, which may be utilized to store and retrieve software programs incorporating computer code that implements the invention. Bak further emphasizes¹ that computer readable storage medium may be in a network including the internet. [Bak, col 4, lines 34-35].

(Page 2, Office Action, 10/20/03).²

It is submitted that the boiler-plate language in *Bak* referred to by the Examiner is intended to support claims directed to a computer readable storage medium that contains code that when executed loads the Java class files 203 using the methods taught in *Bak*. Appellant respectfully submits that *Bak* does not even suggest that the Java class files 203 may be loaded from a computer readable storage medium on a network. Appellant submits that *Bak* would not be motivated to load the Java class files 203 via a network because the computer system of *Bak* is not a platform having limited resources.³ It is further submitted that the Examiner obtained the idea that the Java class files 203 of *Bak* may be loaded from a computer readable storage media on a network not from reading *Bak* but from reading Appellant's disclosure.

B. *Bak* and *Ebrahim* do not disclose or suggest selecting and purging arrays and references from a class structure as claimed in claims 20, 31, and 40.

Appellant submits that *Bak* and *Ebrahim* do not disclose or suggest selecting and purging arrays and references from a class structure as claimed

¹ The emphasis to which the Examiner refers is the following sentence that appears in *Bak*: "Additionally, a data signal embodied in a carrier wave (e.g., in a network including the Internet) may be the computer readable storage medium." (*Bak*, col. 4, lines 43-45).

² The Examiner also cited the existence of the network interface 65 in *Bak* in support of his argument that *Bak* teaches class loading via a network even though the Detailed Description of the Preferred Embodiments in *Bak* does not state a use for the network interface 65 or even refer to it other than to state that it exists.

³ For example, the computer system of *Bak* includes several fixed and removable storage devices. (*Bak*, Figure 2 and col. 4, lines 49-52).

in claims 20, 31, and 40. The Examiner has acknowledged that *Bak* does not disclose a memory manager that selects and purges arrays and references from a class structure. (Page 4, Office Action, 10/20/03).

Ebrahim does not disclose or suggest selecting and purging arrays and references from a class structure as claimed in claims 20, 31, and 40. Instead, *Ebrahim* teaches recovering unused portions of a heap 116. (*Ebrahim*, col. 2, lines 46-49).

Appellant submits that the heap 116 of *Ebrahim* is not a class structure as claimed in claims 20, 31, and 40. Instead, the heap 116 of *Ebrahim* is a repository for storing instances of classes obtained from in a class structure. (*Ebrahim*, col. 3, lines 36-42). For example, *Ebrahim* discloses a class repository 150 for storing classes and states that

in a computer system set up to execute Java bytecode programs, memory 108 will include at least one class repository 150, for locally storing object classes 152 in use and/or available for use by users of the computer system 100. The heap 116 acts as an object repository for storing objects, which are instances of objects of the object classes stored in the class repository 150.

(*Ebrahim*, col. 3, lines 36-42). It is submitted that *Ebrahim* does not even suggest that the class repository 150 itself may be purged of arrays and references as claimed in claims 20, 31, and 40.

C. *Bak* and *Ebrahim* do not disclose or suggest selecting and purging arrays and references from a class structure so as to minimize an amount of memory consumed by the class structure while also minimizing class loading activities on a network as claimed in claims 20, 31, and 40.

Appellant submits that *Bak* and *Ebrahim* do not disclose or suggest selecting and purging arrays and references so as to minimize an amount of memory consumed by a class structure as claimed in claims 20, 31, and 40. Instead, *Ebrahim* teaches a garbage collector that recovers unused portions of a heap 116 (*Ebrahim*, col. 2, lines 46-49) that is not a class structure but instead contains instances of classes obtained from a separate class repository 150 (*Ebrahim*, col. 3, lines 36-42).

Moreover, *Ebrahim* does not even teach minimizing an amount of memory taken up by the heap 116. Instead, *Ebrahim* teaches that space in the heap 116 is recovered so that the tasks that use the heap 116 will not run out of available memory in the heap 116 (*Ebrahim*, col. 2, lines 15-26). It is submitted that the method of garbage collection taught by *Ebrahim* does not reduce the size of the heap 116. Instead, *Ebrahim* teaches a garbage collection method that divides the heap into two halves that are used one at a time and then copies objects in use to one half of the heap. (*Ebrahim*, col. 2, line 50 through col. 3, line 18).⁴

Given that *Bak* and *Ebrahim* do not disclose or suggest obtaining a set of classes via a network while executing an application program, it follows that *Bak* and *Ebrahim* cannot teach or suggest selecting and purging arrays and references from a class structure so as to minimize class loading activities on the network as claimed in claims 20, 31, and 40.

⁴ *Ebrahim* states that a garbage collector “compacts” memory storage. (*Ebrahim*, col 3, line 12). A close reading of *Ebrahim* reveals that the garbage collector compacts storage used by a task so that there are no “holes” between objects used by the task (*Ebrahim*, col 3, line 12-13) rather than reduce the overall size of the heap 116.

II: Claims 23-24, 33-34, and 42-43 are not obvious in view of *Bak* and *Ebrahim* and *Shaughnessy* because *Bak* and *Ebrahim* and *Shaughnessy* do not do not disclose or suggest the limitations of claims 20, 31, and 40.

Appellant submits that claims 23-24, 33-34, and 42-43 are not obvious in view of *Bak* and *Ebrahim* and *Shaughnessy* because claims 23-24, 33-34, and 42-43 depend from claims 20, 31, and 40, respectively, and *Bak* and *Ebrahim* and *Shaughnessy* do not do not disclose or suggest the limitations of claims 20, 31, and 40. Appellant has shown that *Bak* and *Ebrahim* do not disclose or suggest obtaining a set of classes via a network as needed while executing an application program and selecting and purging arrays and references from a class structure so as to minimize an amount of memory consumed by the class structure while also minimizing class loading activities on a network as claimed in claims 20, 31, and 40.

Shaughnessy does not disclose or suggest obtaining a set of classes via a network as needed while executing an application program and selecting and purging arrays and references from a class structure so as to minimize an amount of memory consumed by the class structure while also minimizing class loading activities on a network as claimed in claims 20, 31, and 40. Instead, *Shaughnessy* discloses a development system that enables users to create application programs. (*Shaughnessy*, col. 4, lines 19-23).

III: Claims 29-30 and 38-39 are not obvious in view of *Bak* and *Ebrahim* and *Brown* because *Bak* and *Ebrahim* and *Brown* do not do not disclose or suggest the limitations of claims 20 and 31.

Appellant submits that claims 29-30 and 38-39 are not obvious in view of *Bak* and *Ebrahim* and *Brown* because claims 29-30 and 38-39 depend from claims 20 and 31, respectively, and *Bak* and *Ebrahim* and *Brown* do not do not disclose or suggest the limitations of claims 20 and 31. Appellant has shown that *Bak* and *Ebrahim* do not disclose or suggest obtaining a set of classes via a network as needed while executing an application program and selecting and purging arrays and references from a class structure so as to minimize an amount of memory consumed by the class structure while also minimizing class loading activities on a network as claimed in claims 20 and 31.

Brown does not disclose or suggest obtaining a set of classes via a network as needed while executing an application program and selecting and purging arrays and references from a class structure so as to minimize an amount of memory consumed by the class structure while also minimizing class loading activities on a network as claimed in claims 20 and 31. Instead, *Brown* teaches methods for modifying Java class files to improve execution performance. (*Brown*, col. 3 line 66 through col. 4, line 10).

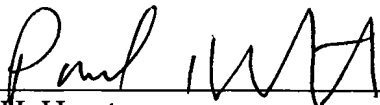
CONCLUSION

Appellant respectfully submits that the stated rejections cannot be maintained in view of the arguments set forth above. Appellant respectfully submits that all of the claims 20-46 are patentable under 35 U.S.C. §103 over the references cited by the Examiner and requests that the Board of Patent Appeals and Interferences direct allowance of the rejected claims.

Respectfully submitted,

By

Date: 7-9-04



Paul H. Horstmann
Reg. No. 36,167

APPENDIX

20. A virtual machine, comprising:
- class loader that enables the virtual machine to obtain a set of classes via a network as needed while executing an application program, the class loader converting the classes obtained via the network into a predefined class definition format and then storing the classes into a class structure in a memory such that the classes stored in the class structure are represented as a set of arrays and references of the predefined class definition format;
- memory manager that selects and purges the arrays and references of the classes from the class structure so as to minimize an amount of the memory consumed by the class structure and to minimize class loading activities on the network.
21. The virtual machine of claim 20, wherein the memory manager deletes a set of objects from the memory which are associated with the classes purged from the class structure.
22. The virtual machine of claim 21, further comprising a list of associations between the objects and the classes stored in the class structure such that the memory manager deletes the objects in response to the list.
23. The virtual machine of claim 20, wherein the memory manager selects a least recently used class in the class structure and purges the arrays and references of the least recently used class from the class structure if an instance of the least recently used class is not being used by the application program.
24. The virtual machine of claim 20, wherein the memory manager selects a least recently used class in the class structure and purges the arrays and

references of the least recently used class from the class structure if an instance of the least recently used class or of a parent class or of a child class of the least recently used class is not being used by the application program.

25. The virtual machine of claim 24, further comprising a list of hierarchical associations among the classes in the class structure such that the memory manager determines whether the instances of the parent class or of the child class are not being used in response to the list.

26. The virtual machine of claim 20, wherein the memory manager purges the classes from the class structure at periodic times.

27. The virtual machine of claim 20, wherein the memory manager purges the classes from the class structure if an amount of available memory falls below a predetermined threshold level.

28. The virtual machine of claim 20, wherein the memory manager purges the classes from the class structure during system idle periods.

29. The virtual machine of claim 20, wherein the class loader obtains the classes from an HTTP server that exports a set of class files containing one or more of the classes.

30. The virtual machine of claim 29, wherein the virtual machine is provided with a class definition statement that specifies one or more URLs for the class files.

31. A method for class loading in a virtual machine, comprising the steps of:

obtaining a set of classes via a network as needed while executing an application program;

converting the classes obtained via the network into a predefined class definition format and then storing the classes into a class structure in a memory such that the classes stored in the class structure are represented as a set of arrays and references of the predefined class definition format;

selecting and purging the arrays and references of the classes from the class structure so as to minimize an amount of the memory consumed by the class structure and to minimize class loading activities on the network.

32. The method of claim 31, further comprising the step of deleting a set of objects from the memory which are associated with the classes purged from the class structure.

33. The method of claim 31, wherein the steps of selecting and purging comprise the steps of:

selecting a least recently used class in the class structure;

determining whether an instance of the least recently used class is being used by the application program;

purging the arrays and references of the least recently used class from the class structure if the instance is not being used.

34. The method of claim 33, wherein the step of determining whether an instance of the least recently used class is being used comprises the step of determining whether an instance of the least recently used class or of a parent class or of a child class of the least recently used class is being used by the application program.

35. The method of claim 31, wherein the steps of selecting and purging comprise the steps of selecting and purging the classes from the class structure at periodic times.

36. The method of claim 31, wherein the steps of selecting and purging comprise the steps of selecting and purging the classes from the class structure if an amount of available memory falls below a predetermined threshold level.

37. The method of claim 31, wherein the steps of selecting and purging comprise the steps of selecting and purging the classes from the class structure during system idle periods.

38. The method of claim 31, wherein the step of obtaining a set of classes via a network comprises the step of obtaining the classes from an HTTP server that exports a set of class files containing one or more of the classes.

39. The method of claim 31, further comprising the step of providing the virtual machine with a class definition statement that specifies one or more URLs for the class files.

40. A device, comprising:
memory that holds a class structure for storing a set of classes for use when executing an application program;
processor that executes a virtual machine including a class loader that when executed obtains the classes via a network as needed while executing the application program, the class loader converting the classes obtained via the network into a predefined class definition format and then storing the classes into the class structure such that the classes stored in the class structure are represented as a set of arrays and references of the predefined

class definition format, the processor executing a memory manager that selects and purges the arrays and references of the classes from the class structure so as to minimize an amount of the memory consumed by the class structure and to minimize class loading activities on the network.

41. The device of claim 40, wherein the memory manager deletes a set of objects from the memory which are associated with the classes purged from the class structure.

42. The device of claim 40, wherein the memory manager selects a least recently used class in the class structure and purges the arrays and references of the least recently used class from the class structure if an instance of the least recently used class is not being used by the application program.

43. The device of claim 40, wherein the memory manager selects a least recently used class in the class structure and purges the arrays and references of the least recently used class from the class structure if an instance of the least recently used class or of a parent class or of a child class of the least recently used class is not being used by the application program.

44. The device of claim 40, wherein the processor executes the memory manager at periodic times.

45. The device of claim 40, wherein the processor executes the memory manager if an amount of available memory falls below a predetermined threshold level.

46. The device of claim 40, wherein the processor executes the memory manager during system idle periods.